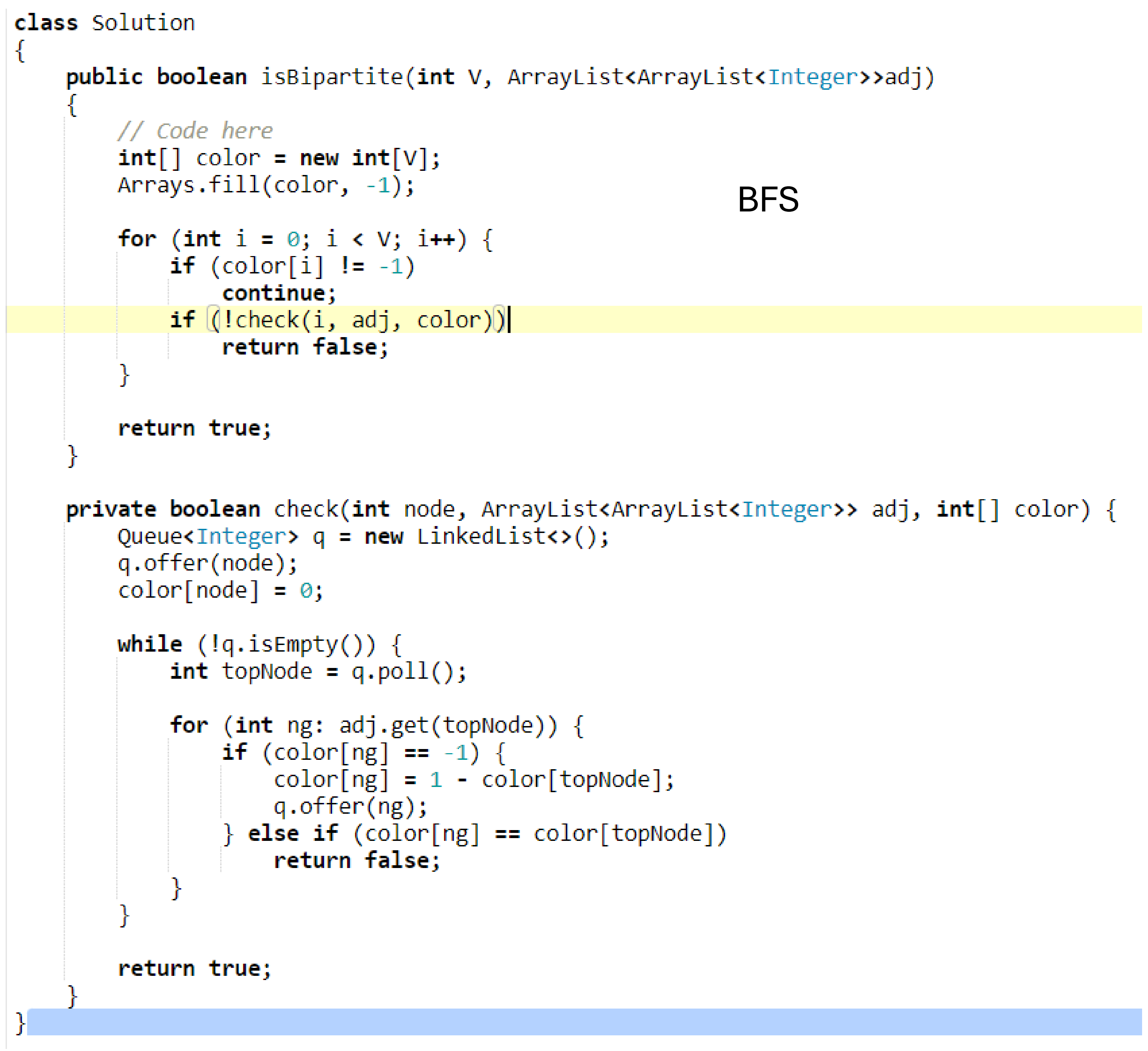
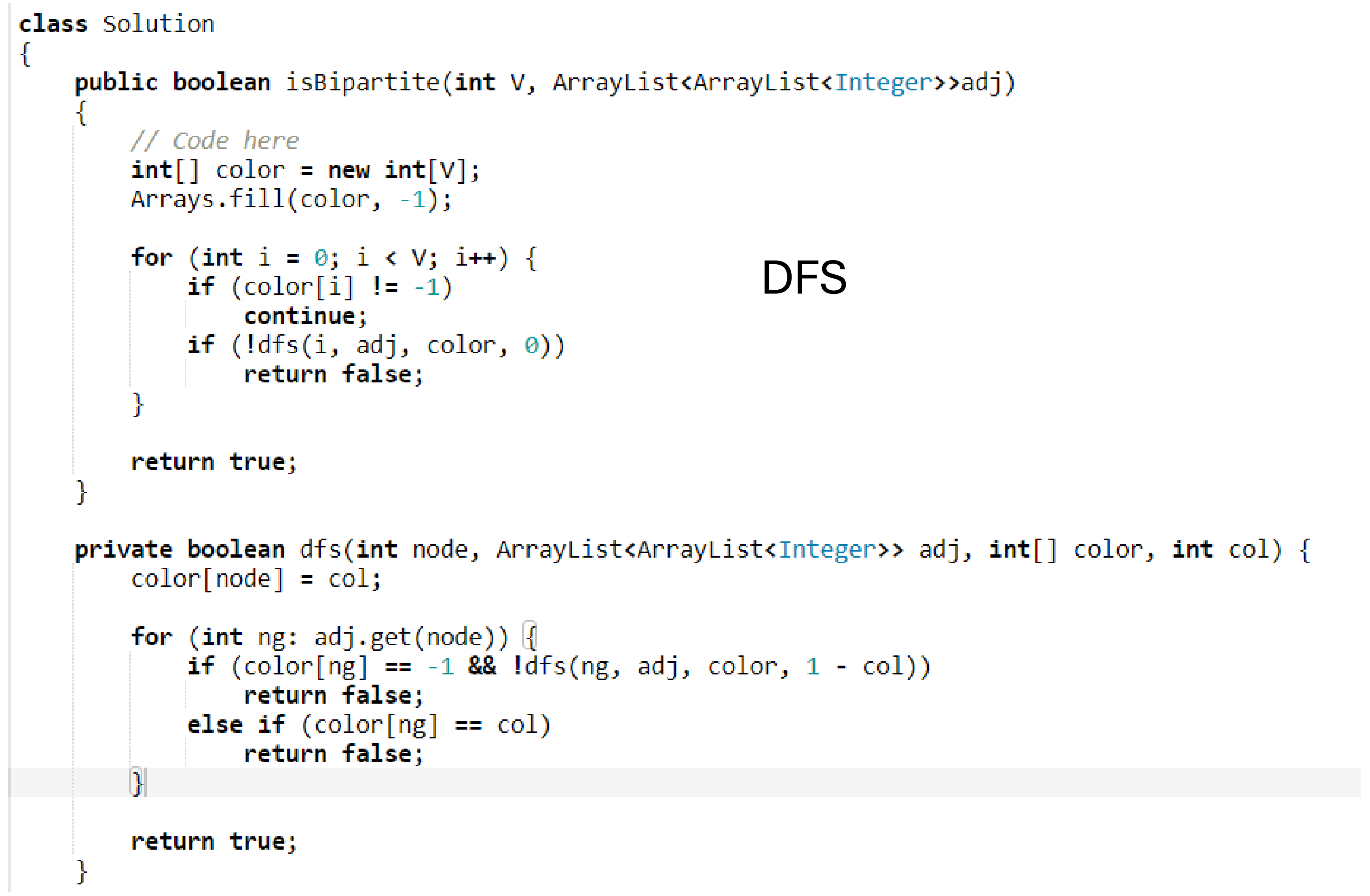
Any graph with odd length cycle cannot be bipartite graph

BFS



```java
class Solution
{
    public boolean isBipartite(int V, ArrayList<ArrayList<Integer>>adj)
    {
        // Code here
        int[] color = new int[V];
        Arrays.fill(color, -1);

        for (int i = 0; i < V; i++) {
            if (color[i] != -1)
                continue;
            if (!check(i, adj, color))
                return false;
        }

        return true;
    }

    private boolean check(int node, ArrayList<ArrayList<Integer>> adj, int[] color) {
        Queue<Integer> q = new LinkedList<>();
        q.offer(node);
        color[node] = 0;

        while (!q.isEmpty()) {
            int topNode = q.poll();

            for (int ng: adj.get(topNode)) {
                if (color[ng] == -1) {
                    color[ng] = 1 - color[topNode];
                    q.offer(ng);
                } else if (color[ng] == color[topNode])
                    return false;
            }
        }

        return true;
    }
}
```

DFS



```java
class Solution
{
    public boolean isBipartite(int V, ArrayList<ArrayList<Integer>>adj)
    {
        // Code here
        int[] color = new int[V];
        Arrays.fill(color, -1);

        for (int i = 0; i < V; i++) {
            if (color[i] != -1)
                continue;
            if (!dfs(i, adj, color, 0))
                return false;
        }

        return true;
    }

    private boolean dfs(int node, ArrayList<ArrayList<Integer>> adj, int[] color, int col) {
        color[node] = col;

        for (int ng: adj.get(node)) {
            if (color[ng] == -1 && !dfs(ng, adj, color, 1 - col))
                return false;
            else if (color[ng] == col)
                return false;
        }

        return true;
    }
```